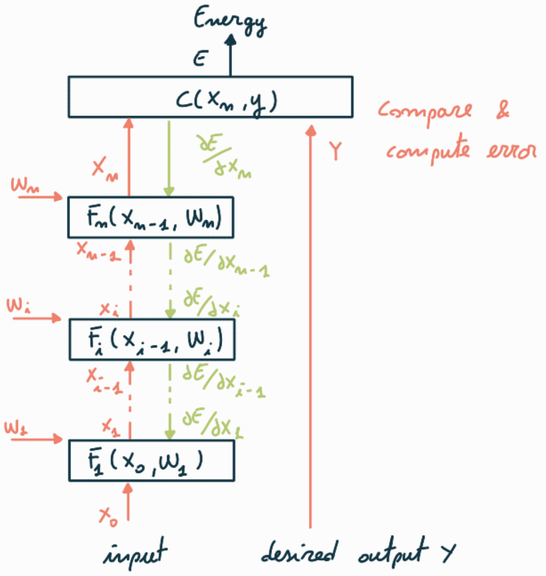
Gradient computation



Information flows forward through the network when computing network output y from input x

⟶

The backpropagation algorithm is used to propagate gradient computation from the cost function through the whole network

⟵

Goal: compute the gradient of the cost function with respect to the parameters:

$$\nabla_\theta J(\theta)$$

Backpropagation is not a training algorithm, it is just an algorithm to estimate the gradient. Training will be done by an optimizer (e.g. Adam) by means of the stochastic gradient descent.

Chain rule

Let $y = g(x)$ and $z = f(g(x)) = f(y)$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x}$$

For vector functions $g: \mathbb{R}^m \to \mathbb{R}^n$ and $f: \mathbb{R}^n \to \mathbb{R}$ we have:

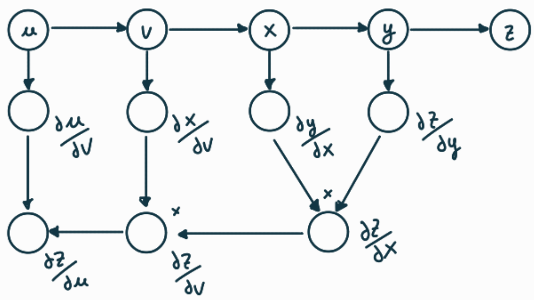
$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j} \frac{\partial y_j}{\partial x_i}$$

In vector notation:

$$\nabla_x z = \left(\frac{\partial y}{\partial x}\right)^T \nabla_y z$$

with $\frac{\partial y}{\partial x}$ the $n \times m$ Jacobian matrix of g

Why is it so easy to compute the gradient?



Estimating gradient ⟶ Backpropagation

Training ⟶ training algorithms ⟶

- Stochastic Gradient Descent (SGD)
- SGD with momentum
- Algorithms with adaptive learning rates

Backpropagation

Backpropagation consists of two steps: ① forward step and ② backward step

① Forward step

Require: Network depth l
Require: $W^{(i)}, i \in \{1, \ldots, l\}$ weight matrices
Require: $\mathbf{b}^{(i)}, i \in \{1, \ldots, l\}$ bias parameters
Require: $\mathbf{x}$ input value
Require: $\mathbf{t}$ target value

$h^{(0)} = \mathbf{x}$
for $k = 1, \ldots, l$ **do**
  $\boldsymbol{\alpha}^{(k)} = \mathbf{b}^{(k)} + W^{(k)}\mathbf{h}^{(k-1)}$
  $\mathbf{h}^{(k)} = f(\boldsymbol{\alpha}^{(k)})$
end for
$\mathbf{y} = \mathbf{h}^{(l)}$
$J = L(\mathbf{t}, \mathbf{y})$

The second step computes the derivative of the error with respect to all the parameters of the network, going backwards from the output layer to the input layer

② Backward step

g is the portion of the gradient that has been computed so far

① $\mathbf{g} \leftarrow \nabla_{\mathbf{y}} J = \nabla_{\mathbf{y}} L(\mathbf{t}, \mathbf{y})$
for $k = l, l-1, \ldots, 1$ **do**
  Propagate gradients to the pre-nonlinearity activations:
② $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\alpha}^{(k)}} J = \mathbf{g} \odot f'(\boldsymbol{\alpha}^{(k)})$ $\{\odot$ denotes elementwise product$\}$

at each layer we need to multiply the contribution of $f(\alpha)$: $f'(\alpha) \cdot$ derivative (α)

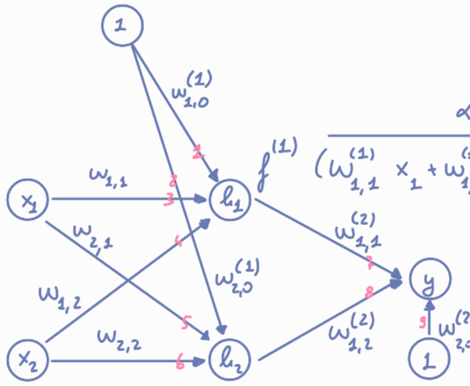
③ $\nabla_{\mathbf{b}^{(k)}} J = \mathbf{g}$
④ $\nabla_{W^{(k)}} J = \mathbf{g}(\mathbf{h}^{(k-1)})^T$
  Propagate gradients to the next lower-level hidden layer:
⑤ $\mathbf{g} \leftarrow \nabla_{\mathbf{h}^{(k-1)}} J = (W^{(k)})^T \mathbf{g}$
end for

$\alpha_i = w_i^T h_{i-1} + b_i$

$\frac{\partial \alpha_i}{\partial w_i} = h_{i-1} \qquad \frac{\partial \alpha_i}{\partial b_i} = 1$

This is not the training, it is just a part of it

Example



$$f^{(1)}\big(\overbrace{w^{(1)}_{1,1}x_1 + w^{(1)}_{1,2}x_2 + w^{(1)}_{1,0}}^{\alpha_1^{(1)}}\big)$$

$$f^{(2)}\big(\underbrace{w^{(2)}_{1,1}h_1 + w^{(2)}_{1,2}h_2 + w^{(2)}_{2,0}}_{\alpha_1^{(2)}}\big)$$

$$f^{(1)}\big(\underbrace{w^{(1)}_{2,1}x_1 + w^{(1)}_{2,2}x_2 + w^{(1)}_{2,0}}_{\alpha_2^{(1)}}\big)$$

we want to compute 9 derivatives

$f^{(1)}(z) = ReLU$
$f^{(2)}(z) = z$
Loss $= MSE$

Forward step

- Given $x_1, x_2, w^{(k)}_{i,j}, t$
- Compute $\alpha_1^{(1)}, \alpha_2^{(1)}, \alpha^{(2)}, h_1^{(1)}, h_2^{(1)}, h^{(2)}, y, J = L(t,y)$

layer 1

$$\alpha_1^{(1)} = w^{(1)}_{1,0} + w^{(1)}_{1,1}x_1 + w^{(1)}_{1,2}x_2 \qquad h_1^{(1)} = f(\alpha_1^{(1)})$$

$$\alpha_2^{(1)} = w^{(1)}_{2,0} + w^{(1)}_{2,1}x_1 + w^{(1)}_{2,2}x_2 \qquad h_2^{(1)} = f(\alpha_2^{(1)})$$

$$\alpha_i^{(1)} = w^{(1)}_{i,0} + w^{(1)}_{i,1}x_1 + w^{(1)}_{i,2}x_2$$
$$h_i^{(1)} = f^{(1)}(\alpha_i^{(1)})$$

layer 2

$$\alpha_1^{(2)} = w^{(2)}_{1,0} + w^{(2)}_{1,1}h_1 + w^{(2)}_{1,2}h_2 \qquad h_1^{(2)} = f(\alpha_1^{(2)}) \qquad y = h^{(2)}$$

Let's use the compact notation

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad W^{(1)} = \begin{bmatrix} w^{(1)}_{1,1} & w^{(1)}_{1,2} \\ w^{(1)}_{2,1} & w^{(1)}_{2,2} \end{bmatrix} \quad b^{(1)} = \begin{bmatrix} w^{(1)}_{1,0} \\ w^{(1)}_{2,0} \end{bmatrix} \quad W^{(2)} = \begin{bmatrix} w^{(2)}_{1,1} & w^{(2)}_{1,2} \end{bmatrix} \quad b^{(2)} = \begin{bmatrix} w^{(2)}_{1,0} \end{bmatrix}$$

$$h^{(0)} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = x$$

layer 1

$$\alpha^{(1)} = \begin{bmatrix} \alpha_1^{(1)} \\ \alpha_2^{(1)} \end{bmatrix} = W^{(1)}h^{(0)} + b^{(1)} \qquad h^{(1)} = \begin{bmatrix} h_1^{(1)} \\ h_2^{(1)} \end{bmatrix} = f^{(1)}(\alpha^{(1)}) = \begin{bmatrix} f^{(1)}(\alpha_1^{(1)}) \\ f^{(1)}(\alpha_2^{(2)}) \end{bmatrix} = \begin{bmatrix} ReLU(\alpha_1^{(1)}) \\ ReLU(\alpha_2^{(1)}) \end{bmatrix}$$

layer 2

$$\alpha^{(2)} = \begin{bmatrix} \alpha_1^{(2)} \end{bmatrix} = W^{(2)}h^{(1)} + b^{(2)} \qquad h^{(2)} = \begin{bmatrix} h_1^{(2)} \end{bmatrix} = f^{(2)}(\alpha^{(2)}) = \begin{bmatrix} \alpha_1^{(2)} \end{bmatrix}$$

$$\alpha^i = W^i h^{i-1} + b^i$$
$$h^i = f^i(\alpha^i)$$

Compute loss function (MSE) $L(t,y) = \frac{1}{2}(t-y)^2$

Backward step

layer 2

① $$g \longleftarrow \nabla_{h^{(2)}} J = \nabla_y J = \nabla_y \frac{1}{2}(t-y)^2 = \frac{1}{2}\, 2\,(t-y)^1(-1) = -(t-y) = y-t$$

② $$g \longleftarrow \nabla_{a^{(2)}} J = g \odot f^{(2)'}(a^{(2)}) = g \odot \frac{\partial a^{(2)} - t}{\partial a^{(2)}} = g \odot 1 = g$$ linear activation function

we computed the gradient of J with respect to $a^{(2)}$; now we want to know the influence of the weights and biases on $\partial J/\partial a^{(2)}$; this will be useful later (SGD) to update them in a training algorithm.

③ $$\nabla_{b^{(2)}} J \longleftarrow \begin{bmatrix} \partial J/\partial w^{(2)}_{1,0} \end{bmatrix} = g$$

④ $$\nabla_{W^{(2)}} J \longleftarrow \begin{bmatrix} \dfrac{\partial J}{\partial w^{(2)}_{1,1}} & \dfrac{\partial J}{\partial w^{(2)}_{1,2}} \end{bmatrix} = g \cdot (h^{(1)})^T$$

$$W^{(2)} = \begin{bmatrix} w^{(2)}_{1,1} & w^{(2)}_{1,2} \end{bmatrix}$$

Propagate gradients to the next lower-level hidden layer

⑤ $$g \longleftarrow \nabla_{h^{(k-1)}} J = \nabla_{h^{(1)}} J = (W^{(k)})^T g = (W^{(2)})^T g = \begin{bmatrix} w^{(2)}_{1,1} & w^{(2)}_{1,2} \end{bmatrix}^T g$$

layer 1

② $$g \longleftarrow \nabla_{a^{(1)}} J = g \odot f^{(1)'}(a^{(1)}) = g \odot \begin{bmatrix} \partial \mathrm{ReLU}(a^{(1)}_1)/\partial a^{(1)}_1 \\ \partial \mathrm{ReLU}(a^{(1)}_2)/\partial a^{(1)}_2 \end{bmatrix} = g \odot \begin{bmatrix} \mathrm{step}(a^{(1)}_1) \\ \mathrm{step}(a^{(1)}_2) \end{bmatrix}$$

③ $$\nabla_{b^{(1)}} J \longleftarrow \begin{bmatrix} \partial J/\partial w^{(1)}_{1,0} \\ \partial J/\partial w^{(1)}_{2,0} \end{bmatrix} = g$$

④ $$\nabla_{W^{(1)}} J \longleftarrow \begin{bmatrix} \partial J/\partial w^{(1)}_{1,1} & \partial J/\partial w^{(1)}_{1,2} \\ \partial J/\partial w^{(1)}_{2,1} & \partial J/\partial w^{(1)}_{2,2} \end{bmatrix} = g\,(h^{(1)})^T$$